Implementasi *Network Monitoring System* Menggunakan *Librenms* Berbasis *Docker Container*

Alfian Tegar Putra Afandi
D3 Manajemen Informatika, Fakultas Teknik, Universitas Negeri Surabaya
alfian.17050623015@mhs.unesa.ac.id

Asmunin
Teknik Informatika, Fakultas Teknik, Universitas Negeri Surabaya
asmunin@unesa.ac.id

Abstrak

Dalam sebuah instansi, jaringan selalu dibutuhkan untuk memenuhi kebutuhan transfer data dan sinkronasi dari setiap perangkat yang dipakai. Terutama pada era pandemi covid-19 hampir segala kegiatan dilakukan secara daring menyebabkan peningkatan penggunaan jaringan pada instansi. Hal ini tentu harus didukung dengan pemantauan secara intensif terhadap perangkat jaringan pada sebuah instansi untuk menjaga layanan tersebut agar tetap stabil saat digunakan oleh pengguna. Salah satu hal yang dapat dilakukan adalah penerapan *Network Monitoring System* yang bertujuan untuk memantau kondisi pada perangkat jaringan apakah dalam keadaan baik ataupun sebaliknya dengan bantuan aplikasi *Librenms*. *Librenms* merupakan aplikasi pemantau jaringan yang bersifat *open source* dan memiliki banyak fitur. Seperti: dapat melakukan pemindaian perangkat jaringan secara otomatis, pemantauan terhadap kondisi trafik penggunaan data, pemantauan kondisi pada perangkat jaringan yang direpresentasikan menggunakan *Rrdtool* sebagai pembuat grafik. Sistem ini juga dapat berjalan pada *platform docker* yang didalamnya terdapat aplikasi *librenms* beserta aplikasi pendukung lainnya yang dikonfigurasi dan dijalankan secara bersamaan menggunakan *docker compose*. Dengan adanya docker, aplikasi yang dijalankan dapat berjalan secara terisolasi dalam sebuah *container* tanpa memerlukan hypervisor dan OS tambahan, sehingga dapat mengurangi penggunaan sumber daya komputer server dalam menjalankan virtualisasi. Hasil pengujian membuktikan bahwa aplikasi *librenms* yang dijalankan menggunakan *docker* dapat berjalan dengan baik serta dapat melakukan pemantauan kondisi trafik data dan pemantauan kondisi *resource*, dan juga mampu melakukan sistem peringatan pada perangkat yang memudahkan *administrator* melakukan pemantauan jaringan.
Kata Kunci: *Network Monitoring System, Librenms,* SNMP, *Docker*.

Abstract

A network is always needed in an institution to meet the needs of data transfer and synchronization of every device used. Especially in the era of the covid 19 pandemics, almost all activities carried out online led to an increase in the use of networks in institutions. This must be supported by intensive monitoring of network devices in an institution to keep the service stable when used by users. One of the things that can be done is the implementation of a Network Monitoring System which aims to monitor the condition on the network device whether in good condition or vice versa with the help of a Librenms application. Librenms is an open-source network monitoring application that has many features. This of course must be supported by intensive monitoring of network devices in an institution to keep the service stable when used by users. One of the things that can be done is the implementation of a Network Monitoring System which aims to monitor the condition of network devices whether they are in good condition or otherwise with the help of the Librenms application. Librenms is a network monitoring application that is open source and has many features. Such as: can perform an automatic scan of network devices, monitor data usage traffic conditions, and monitor conditions on network devices represented using Rrdtool as a graph maker. The system can also run on docker platforms where Librenms applications and other supporting applications are configured and run simultaneously using docker-compose. With docker, applications that run can run isolated in a container without the need for a hypervisor and additional OS, thereby reducing the use of server computer resources in running virtualization. The test results show that the librenms application that is run using docker can run well and can monitor data traffic conditions and monitor resource conditions, and is also able to perform a warning system on devices that make it easier for administrators to monitor network.
Keyword: *Network Monitoring System, Librenms, SNMP, Docker*.

PENDAHULUAN

Dunia semakin maju dengan adanya perkembangan teknologi dan jaringan. Perkembangan tersebut memudahkan komunikasi antara satu dengan lainnya yang dibangun sesuai dengan kebutuhan institusi maupun kebutuhan pribadi. Dalam sebuah instansi, jaringan selalu dibutuhkan untuk memenuhi kebutuhan transfer data dan sinkronasi dari setiap perangkat yang dipakai. Terlebih lagi di tengah pandemi COVID-19 memberikan dampak bagi semua kalangan, ditambah lagi kebijakan yang diputuskan oleh pemerintah untuk menghimbau segala kegiatan dilakukan dari rumah atau dengan istilah *WFH*(*Work From Home*) (Yahya:2021).

Dengan diterapkannya *WFH*, maka peningkatan kinerja sumber daya perangkat jaringan pada sebuah instansi sangat diperlukan dan perlunya pengawasan secara intensif agar layanan tetap stabil. Oleh sebab itu, dibutuhkan juga sistem untuk melakukan pemantauan terhadap perangkat jaringan. Kegiatan pemantauan jaringan dengan tujuan memantau kondisi dan kinerja pada sebuah perangkat jaringan yang dilakukan dalam sebuah sistem atau biasa disebut dengan *Network Monitoring System.*

Saat ini sudah banyak aplikasi yang dapat memantau perangkat jaringan. Salah satunya aplikasi *Librenms Docker* yang merupakan aplikasi pemantau jaringan dikembangkan oleh *Librenms* dan sifatnya *open source*, aplikasi ini berjalan di dalam *docker container* dan menggunakan protokol SNMP yang berfungsi untuk memonitor beberapa perangkat jaringan. Dengan adanya Docker, aplikasi yang dijalankan dapat berjalan secara terisolasi dalam sebuah *container* tanpa memerlukan hypervisor dan OS tambahan, sehingga dapat mengurangi penggunaan sumber daya komputer server dalam menjalankan virtualisasi. Selain itu, Docker juga mengikat aplikasi beserta konfigurasi dalam *container* yang dapat diubah ke bentuk *docker image*, sehingga aplikasi dapat langsung digunakan berkali-kali tanpa perlu mengatur ulang konfigurasi. Aplikasi *Librenms* memiliki keunggulan pada fitur-fiturnya seperti fitur *auto discovery, alerting, Service Monitoring (Nagios plugins), Device Backup Integration* (Saputra, Wiharta & Sastra:2020). Serta dalam hasil pemantauannya terkelola oleh *rrdtools* yang merepresentasikan data pemantauan jaringan dalam bentuk grafik, sehingga hasil pemantauan dapat dilihat secara jelas.

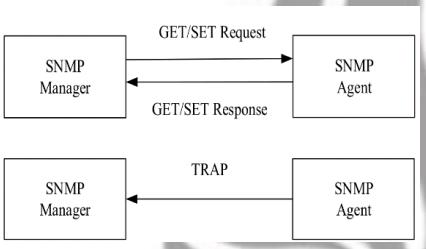
Penelitian sebelumnya dengan judul implementasi sistem pemantauan jaringan menggunakan *librenms* pada jaringan kampus Universitas Udayana membahas tentang implementasi pemantauan jaringan pada salah satu instansi menggunakan *Librenms*(Saputra, Wiharta & Sastra:2020). Kemudian pada penelitian ini lebih mengacu pada implementasi *Librenms Docker* yang dimana pada proses konfigurasi *deployment* ini nantinya berjalan menggunakan *docker container* sebagai wadah dari beberapa *dependendcy* pada komponen yang dibutuhkan dalam menjalankan aplikasi *Librenms Docker*.

Berdasarkan latar belakang diatas, penulis bertujuan melanjutkan penelitian yang selanjutnya ditulis dalam bentuk tugas akhir dengan judul "**IMPLEMENTASI *NETWORK MONITORING SYSTEM MENGGUNAKAN LIBRENMS BERBASIS DOCKER***".

KAJIAN PUSTAKA

Simple Network Management Protocol (SNMP)

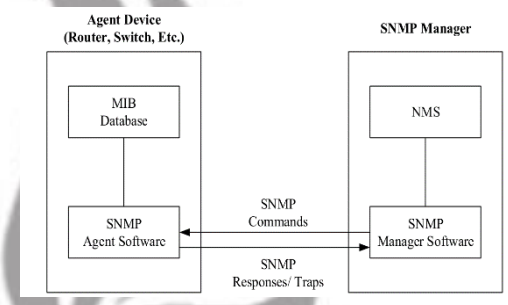
Menurut Pradikta Reza (2013:155), *Simple Network Management Protocol* atau yang disingkat SNMP adalah protokol jaringan TCP/IP yang menangani manajemen jaringan agar dapat dengan mudah memantau kondisi jaringan. SNMP menyediakan beberapa opsi yang dapat mengumpulkan data, memantau beberapa perangkat jaringan, mengkonfigurasi, memanipulasi serta dapat bertukar informasi antar perangkat yang ada pada satu jaringan.



Gambar 1. Model Simulasi SNMP

Dengan protokol SNMP, manajemen dan pemantauan jaringan dapat dilakukan secara *remote*. Artinya perangkat dapat dipantau dari jarak jauh tanpa harus ke lokasi perangkat terlebih dahulu (Saputra, 2020). Hal ini tentu memudahkan proses audit dan konfigurasi perangkat karena dapat dilakukan kapanpun dan dimanapun.

Terdapat tiga komponen utama pada SNMP, yaitu SNMP Manager, SNMP Agent, dan MIB (*Management Information Base*) (Segara, 2018). SNMP Manager berjalan pada host yang untuk berkomunikasi dengan agent untuk meminta informasi jaringan. Sementara SNMP Manager mengumpulkan informasi dari agent, SNMP Agent akan mendeteksi kondisi jaringan pada setiap perangkat yang terpasang (Pratama, 2017). SNMP Agent terpasang pada *router*, *switch*, PC, dan perangkat jaringan lain yang saling terhubung satu sama lain. Informasi tentang aktivitas perangkat ini nanti akan dikirimkan ke SNMP Manager. Informasi ini nantinya disimpan dalam sebuah *database* utama SNMP yang disebut MIB. (Segara, 2018). *Database* ini diisi oleh SNMP manager berupa informasi yang didapatkan dari SNMP agent. Struktur MIB berupa diagram pohon yang menempatkan setiap *Object Identifier* (OID) pada setiap pohon. Berikut ini adalah struktur diagram dari SNMP.



Gambar 2. Diagram Struktur SNMP

Network Monitoring System (NMS)

Network Monitoring System (NMS) adalah perangkat lunak yang digunakan untuk melakukan monitoring pada elemen-elemen dalam jaringan komputer seperti *router*, *switch* maupun *server* (Hidayat, 2020). NMS umumnya menggunakan protokol SNMP yang dirancang untuk mengumpulkan data manajemen perangkat jaringan dan konfigurasi perangkat secara *real-time* dari jarak jauh. Hasil dari pemantauan tersebut menjadi bahan pertimbangan dalam pengambilan keputusan oleh pihak manajemen maupun sebagai bahan analisa apakah terdapat kejanggalan dalam operasional jaringan.

Banyak perangkat lunak NMS yang dikembangkan secara *open-source* sehingga dapat

digunakan secara gratis. Perangkat lunak NMS yang dapat digunakan secara gratis antara lain Zabbix, OpenNMS, *LibreNMS*, Cacti, dan lain-lainnya.

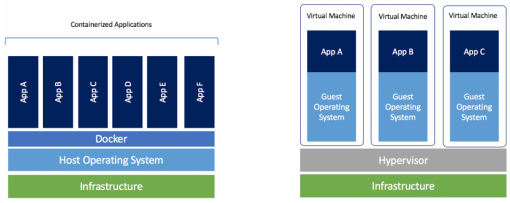
Docker

Pada saat ini, pengembangan aplikasi dan *server* menjadi sangat cepat seiring perkembangan komputer dan internet. Dalam pengembangan aplikasi ini dibutuhkan teknologi virtualisasi yang dapat membangun banyak *server* dan mencegah masalah perbedaan versi pada program, seperti perbedaan *library* serta *tools* yang digunakan. Dengan meningkatnya jumlah aplikasi yang dikembangkan, teknologi *virtual machine* pada virtualisasi dirasa kurang efektif karena penggunaan sumber daya yang cenderung besar dan kurang efisien. Teknologi *Container* muncul sebagai jawaban dari masalah ini.

Docker merupakan platform pengembangan aplikasi yang menggunakan teknologi *Container* (Apridayanti, 2018). *Docker* pertama kali dikembangkan oleh Solomon Hykes bersama Andrea Luzzardi dan Francois-Xavier Bourlet dalam sebuah project internal bernama DotCloud. Pengembangan *Docker* dilakukan pertama kali pada tahun 2009 dengan nama DotCloud Inc. Kemudian project ini berganti nama menjadi *Docker* pada 2013 hingga sekarang (Harfiansyah, 2016).

Berbeda dengan virtualisasi dimana aplikasi akan berjalan diatas hypervisor dan Host OS, *Docker* dapat menjalankan aplikasi tanpa hypervisor dan Host OS sehingga dapat meningkatkan efisiensi sumber daya yang digunakan aplikasi.

Selain dapat berjalan tanpa memerlukan Hypervisor dan OS tambahan, *Docker* juga dapat 'membungkus' sebuah sebuah aplikasi beserta library dan file konfigurasinya dalam bentuk *Docker Image*. Menurut situs resmi *Docker*, developer dapat memangkas proses pengembangan aplikasi mereka dikarenakan tidak perlu melakukan konfigurasi yang sama pada aplikasi yang mereka kembangkan secara berulang-ulang.



Gambar 3. Model Infrastruktur Docker (kiri) vs Virtualisasi (kanan)

***LibreNMS* Docker**

LibreNMS Docker adalah sebuah *Docker Image* yang dikembangkan oleh *LibreNMS project* (http://www.librenms.org/). *Image* ini berisi *bundle* dari aplikasi *LibreNMS* yang berbasis Alpine Linux dan *web server Nginx* (baca : *Engine X*). *LibreNMS* Docker *Image* dikembangkan dibawah lisensi MIT yang bersifat *open-source*, sehingga *Image* ini dapat digunakan dan dimodifikasi secara gratis.

Dengan *Image LibreNMS* ini, pengguna dapat melakukan monitoring beberapa *container* lain yang juga berjalan pada Docker. Hal ini dikarenakan *LibreNMS* adalah *container* yang bekerja secara *side-car*, artinya *container* ini dapat bekerja bersamaan dengan *container* lain secara terintegrasi. Hal ini tentu dapat menghemat jumlah perangkat jaringan yang digunakan karena dalam melakukan monitoring, pengguna tidak perlu menyediakan perangkat lain untuk me-monitor perangkat jaringannya.

Selain itu, *Image LibreNMS* ini juga bersifat *agnostic*, artinya *image* ini dapat berjalan pada berbagai macam tipe OS yang tersedia, tanpa terikat

pada OS tertentu. Hal ini dapat memudahkan pengguna karena tidak perlu memilih OS tertentu untuk menjalankan *image LibreNMS* serta mempermudah proses *migrating* apabila pengguna ingin memindahkan *image* ini pada perangkat jaringan atau *server* lain.

Nginx

Nginx (Baca : Engine X) adalah sebuah *web server* yang berjalan pada protokol HTTP (Nginx Wiki). *Web server* ini pertama kali dikembangkan oleh Igor Sysoev yang berkebangsaan Rusia pada tahun 2004. *Web server* ini bersifat *open-source* sehingga dapat dipakai dan dimodifikasi secara gratis. Nginx dibangun menggunakan PHP sebagai bahasa pemrograman utamanya.

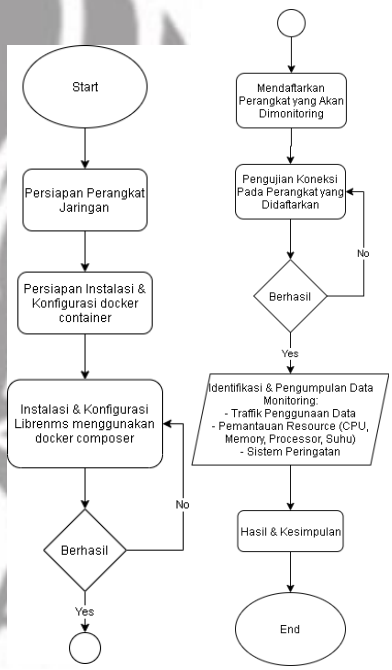
Selain sebagai *web server*, Nginx juga dapat digunakan sebagai *reverse proxy*, yang digunakan untuk menentukan pengiriman *request* dari *client* dan *server*. Nginx cukup dikenal dengan performa yang baik, stabilitas yang tinggi, konfigurasi yang sederhana, serta banyaknya fitur yang didukung. Hal ini membuat Nginx menjadi salah satu *web server* paling populer di dunia (opensource.com, 2016).

METODE PENELITIAN

Analisis Sistem

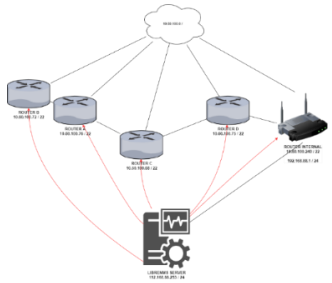
Flowchat dibawah merupakan alur kerja pada sistem monitoring yang akan dibuat serta penjelasan alurnya sebagai berikut :

1. Menyiapkan beberapa perangkat jaringan seperti *Router*, *switch*, sistem operasi linux yang digunakan sebagai *server* pada *librenms*, dan sistem operasi windows yang digunakan sebagai client untuk mengakses aplikasi *librenms*.
2. Melakukan konfigurasi dan instalasi pada *Docker container*.
3. mengkonfigurasi aplikasi librenms serta melakukan *deployment* pada komputer *server* yang digunakan untuk melakukan pemantauan jaringan menggunakan *docker compose*.
4. Mendaftarkan alamat IP pada perangkat yang akan dimonitoring serta membuat konfigurasi SNMP *agent*.
5. Uji coba koneksi pada perangkat yang akan dimonitoring.
6. Mengidentifikasi dan mengumpulkan data pemantauan jaringan.



Gambar 4. Alur Perancangan Sistem

Pemantauan disimulasikan di Laboratorium Jaringan Komputer gedung A10 Universitas Negeri Surabaya. Terdapat beberapa penjelasan terkait dengan topologi yang digambarkan antara lain sebagai berikut:



Gambar 5. Topologi jaringan pada sistem monitoring

1. Terdapat beberapa *router* dan satu *router switch* yang memiliki rentang alamat *IP* 10.60.100.0/24 sebagai perangkat SNMP *agent*.
2. *Router switch* memiliki alamat *IP* 192.168.88.1/24 sebagai jaringan internal yang menghubungkan dengan komputer *server librenms.*
3. Komputer *server* dengan alamat *IP* 192.168.88.253/22 yang didalamnya terdapat aplikasi *librenms* bertugas untuk memantau setiap perangkat jaringan yang terhubung dengan SNMP *agent*.

Spesifikasi Kebutuhan Perangkat

Spesifikasi komputer yang digunakan untuk melakukan pemantauan jaringan menggunakan *docker librenms* dapat dijelaskan pada tabel berikut :

Sistem Operasi	Ubuntu Desktop 20.04 LTS 64-bit
Processor	Intel® Core™ i7-7700T CPU @ 2.90GHz
RAM	4.00 GB
Kapasitas	HDD 1 TB
VGA	NV118 / Mesa Intel® HD Graphics 630 (KBL GT2)

Tabel 1. Spesifikasi komputer

Spesifikasi Kebutuhan Perangkat Lunak

Terdapat beberapa perangkat lunak yang dibutuhkan untuk menjalankan aplikasi *librenms* yang berjalan pada *docker container* dapat dijelaskan pada tabel berikut :

Perangkat Lunak	Keterangan
Docker Container	Sebagai penampung beberapa *docker image* yang akan dikonfigurasi serta dijalankan dalam *container* tersebut.
Docker Compose	Sebuah *tools* yang mengelola setiap *container* pada docker dijalankan dalam satu *environment* yang telah dikonfigurasi dalam sebuah *file* berformat YAML.

Tabel 2. Spesifikasi kebutuhan perangkat lunak

HASIL DAN PEMBAHASAN

Pada tahapan ini membahas hasil dari instalasi dan konfigurasi aplikasi *librenms* yang menggunakan *docker container* pada *server* serta hasil pemantauan pada perangkat jaringan.

Konfigurasi Docker

Dalam implementasi ini akan melakukan instalasi pada platform *server* Ubuntu dan melakukan instalasi *dependency* yang sudah tersedia di *package manager* pada OS dengan memasukkan perintah pada gambar dibawah ini.

```
: $ sudo apt-get install docker.io
```

Gambar 6. Perintah instalasi docker

Untuk memastikan bahwa *docker* telah terinstal pada *server* dapat dilihat dengan memasukkan

perintah "*docker -v*" maka akan terdapat beberapa keterangan seperti gambar berikut.

```
johnsorg0n@linux:~$ docker -v
Docker version 20.10.2, build 20.10.2-0ubuntu1~20.04.2
johnsorg0n@linux:~$
```

Gambar 7. Perintah untuk mengecek versi Docker

Agar *docker* dapat dijalankan tanpa menggunakan *root privilege* maka harus memasukkan perintah seperti gambar berikut.

```
johnsorg0n@linux:~$ sudo usermod -aG docker $USER
[sudo] password for johnsorg0n:
johnsorg0n@linux:~$ sudo chmod 666 /var/run/docker
docker/      docker.pid   docker.sock
johnsorg0n@linux:~$ sudo chmod 666 /var/run/docker.sock
johnsorg0n@linux:~$
```

Gambar 8. Konfigurasi untuk mengubah hak akses pada docker

Konfigurasi *Docker Compose*

Tahap selanjutnya yaitu melakukan instalasi pada *docker compose* agar dapat menjalankan banyak *container* menjadi satu *environment*, dengan demikian konfigurasi pada setiap *container* dikemas dan dijalankan secara bersamaan.. Berikut perintah untuk instalalsi *docker compose.*

```
$ sudo apt install docker-compose
```

Gambar 9. Perintah instalasi docker-compose

Konfigurasi *Librenms docker container*

Setelah instalasi *docker compose* berjalan dengan baik, selanjutnya membuat *file* konfigurasi berformat YAML yang didalamnya terdapat beberapa *container* beserta variabel *environment*-nya.

Selanjutnya membuat konfigurasi *file* .env yang didalamnya terdapat nama *database*, *username* dan *password* kemudian dipanggil oleh file berformat YAML. File YAML ini berisi konfigurasi untuk *Librenms Docker* yang dapat digunakan kembali. Sehingga, karena Docker dapat menyimpan file konfigurasi ini pada *container.*, file ini dapat digunakan saat diperlukan instalasi *Librenms Docker* yang baru tanpa harus menulis ulang. Pada gambar 10 merupakan konfigurasi *file* .env.

```
johnsorg0n@linux:~/docker-librenms$ cat .env
TZ=Asia/Jakarta
PUID=1000
PGID=1000

MYSQL_DATABASE=librenms
MYSQL_USER=librenms
MYSQL_PASSWORD=librenms
johnsorg0n@linux:~/docker-librenms$
```

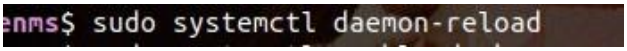
Gambar 10. konfigurasi file .env

Tahapan selanjutnya membuat konfigurasi di dalam *systemd* pada direktori */etc/systemd/system* menggunakan *root privilege* agar *docker compose* dapat mengeksekusi *file* YAML yang telah dibuat dapat dijalankan di latar belakang (*daemonize).* Gambar 11 merupakan konfigurasi pada *docker compose* di dalam *systemd.*

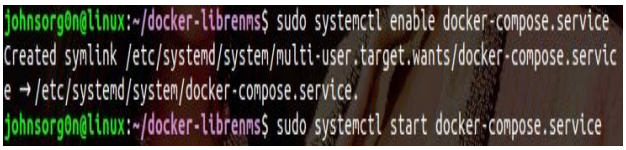
```
[Unit]
Description=Docker Compose Application Service
Requires=docker.service
After=docker.service

[Service]
Type=oneshot
RemainAfterExit=yes
WorkingDirectory=/home/johnsorg0n/docker-librenms/
ExecStart=/usr/bin/docker-compose up -d
ExecStop=/usr/bin/docker-compose down
TimeoutStartSec=0

[Install]
WantedBy=multi-user.target
```

Gambar 11. Konfigurasi systemd untuk menjalankan docker compose

Systemd mengeksekusi program berdasarkan direktori yang telah didefinisikan di dalam *file docker-compose.service* kemudian agar dapat berjalan di latar belakang, dimasukkan perintah seperti pada gambar 12 dan gambar 13**.**



Gambar 12. Perintah systemctl untuk memuat ulang daemon



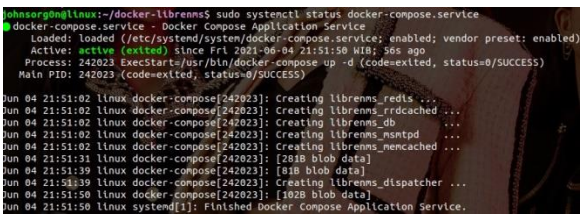
Gambar 13. Perintah untuk mengkaktifkan docker-compose.service

Untuk mengetahui bahwa *service* pada *docker-compose* telah berjalan, dapat dilihat dengan memasukkan perintah "***sudo systemctl status docker-compose***" apabila proses telah berjalan seperti pada gambar 14 akan menampilkan warna hijau yang menunjukkan bahwa *service* telah berjalan pada latar belakang**.**

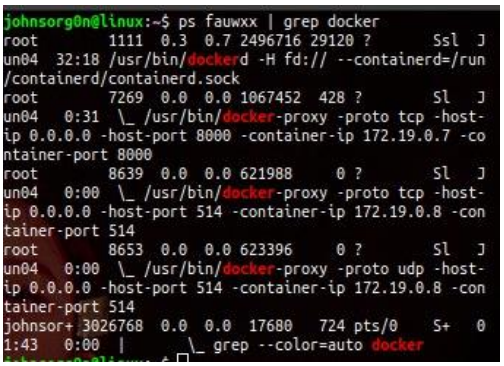
Konfigurasi dan hasil instalasi *librenms* dapat dilihat melalui proses pada *server*. Pada gambar 15 menunjukkan bahwa *docker* telah melakukan *port forwarding* dari *container* dengan alamat IP 172.19.0.7 menuju host melalui *port* 8000.

Konfigurasi dan hasil instalasi *librenms* dapat dilihat melalui proses pada *server*. Pada gambar 15 menunjukkan bahwa *docker* telah melakukan *port forwarding* dari *container* dengan alamat IP 172.19.0.7 menuju host melalui *port* 8000.

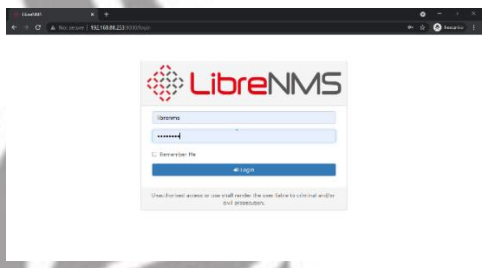
Hasil dari *port forwarding* tersebut diakses melalui *website*. Berikut merupakan halaman awal dari *librenms* berupa tampilan login seperti pada gambar 16**.**



Gambar 14. Perintah untuk mengetahui status pada docker-compose.service



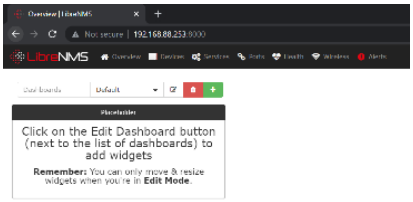
Gambar 15. Melakukan pengecekan terhadap docker pada proses latar belakang



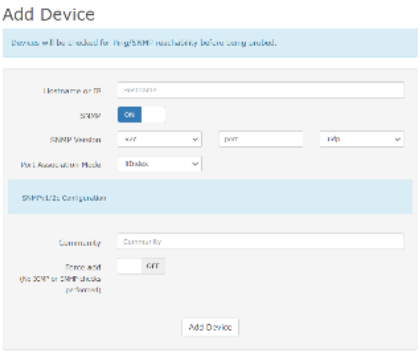
Gambar 16. Tampilan login librenms

Kredensial untuk mengakses *website* berasal dari konfigurasi pada *file* .env yang telah dibuat dengan *username* "*librenms*" dan *password* "*librenms*". Setelah itu maka akan diarahkan ke halaman *dashboard* seperti pada gambar 17.

Penambahan perangkat jaringan dapat dilakukan melalui halaman *Add Device* seperti pada gambar 18**.**

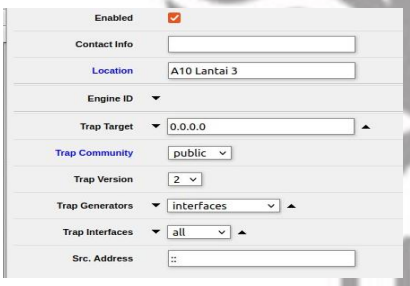


Gambar 17. Halaman tampilan Dashboard



Gambar 18. Halaman tampilan Add device

Pengguna mendaftarkan perangkat jaringan yang akan di *monitoring* dengan memasukkan alamat IP beserta nama *community*-nya. Nama *community* tersebut berasal dari konfigurasi SNMP pada perangkat jaringan. Salah satu contohnya adalah seperti pada gambar 19, *router* membuat nama *community* dengan menggunakan versi SNMP v2c. Setelah itu untuk menguji bahwa SNMP pada perangkat jaringan dapat terhubung dengan *server* dengan memasukkan perintah seperti pada gambar 20.



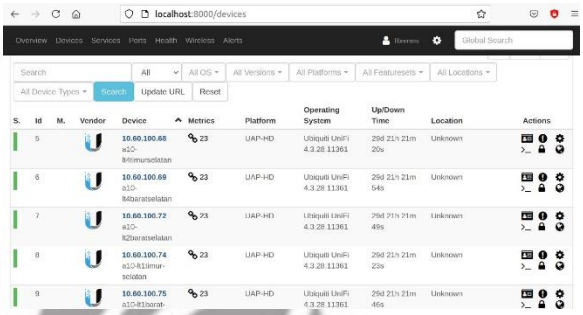
Gambar 19. Konfigurasi SNMP client pada router

```
johnsorg0n@linux:~$ snmpwalk -v2c -c public 10.60.100.77
MIB search path: /home/johnsorg0n/.snmp/mibs:/usr/share/snmp
```

Gambar 20. Perintah untuk menguji SNMP client pada server.

Hasil Konfigurasi *Librenms*

Setelah mengkonfigurasi SNMP pada perangkat jaringan, dapat dilihat pada gambar 21 bahwa perangkat berhasil dimasukkan ke dalam sistem, terdapat beberapa perangkat jaringan diantaranya yaitu *Access Point*, *Router*, dan *Server*. Di dalamnya dilengkapi detail pada perangkat jaringan diantaranya terdapat lokasi, nama sistem operasi, *Up/Down time*, jumlah port yang ada pada setiap perangkat jaringan serta terdapat fitur aksi yang dapat melakukan interaksi berupa *ssh, telnet, http, ping*, dan *ftp* terhadap *client* atau perangkat jaringan yang terdaftar.

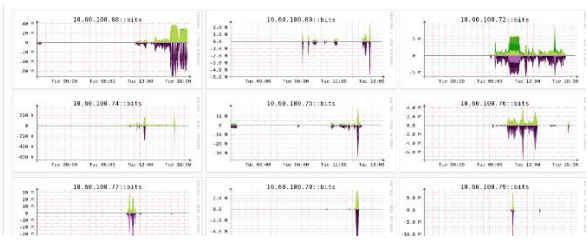


Gambar 21. Halaman tampilan perangkat jaringan yang terdaftar

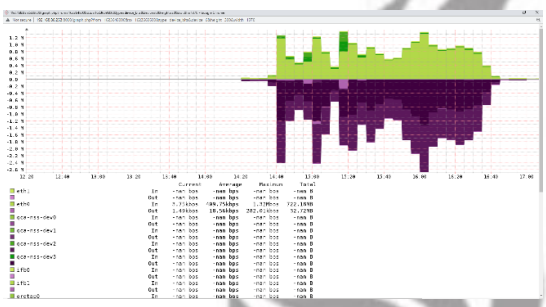
Hasil Pemantauan Trafik Data

Pada perangkat jaringan yang terdaftar, *librenms* mampu melakukan pemantauan trafik *inbound* dan trafik *outbound* disajikan dalam bentuk *graph.* Dapat ditujukkan pada gambar 22 merupakan pemantauan pada beberapa perangkat jaringan dan juga pada gambar 23 yang merupakan detail trafik pada salah satu perangkat jaringan yang dipantau terdapat trafik pada *port* yang dimiliki oleh perangkat tersebut. Pada grafik yang berwarna hijau menunjukkan trafik *inbound* dan yang berwarna ungu merupakan trafik *outbound*. Dari hasil pemantauan trafik data menunjukkan waktu penggunaan rata-rata pada pukul 07.30 hingga 14.00. Tampilan grafik dapat dilihat dalam jangakauan setiap 6 jam, 24 jam, bahkan hingga

tahunan. Serta dapat mengkustomisasi pemantauan sesuai tanggal dan waktu yang ditentukan.



Gambar 22. Halaman Tampilan Pemantauan Trafik data pada keseluruhan perangkat yang terdaftar.



Gambar 23. Halaman tampilan detail pemantauan trafik data.

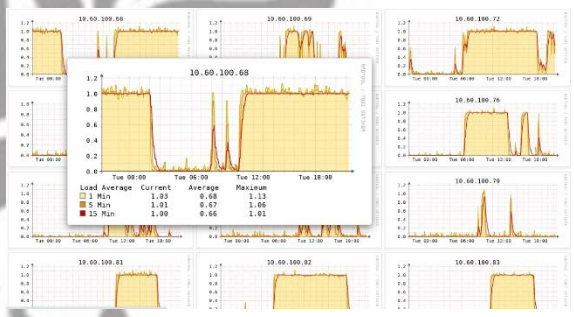
Grafik yang terdapat di setiap fitur merupakan hasil representasi dari *RRDTool* yang merupakan fitur pada *librenms* dan memiliki cara kerja dengan mengolah data pemantauan jaringan menjadi dalam bentuk grafik.

Hasil Pemantauan *Resource*

Librenms selain memiliki kemampuan untuk melakukan pemantauan pada trafik penggunaan data, dapat juga melakukan pemantauan kondisi pada perangkat jaringan yang meliputi pemantauan *CPU, Memory* serta suhu pada perangkat.

Pada gambar 24 adalah tampilan pemantauan CPU, data yang diambil ditampilkan dalam bentuk grafik sehingga dapat mengetahui penggunaan CPU secara berkala, rata-rata penggunaan CPU / *Processor* menggunakan 2% dari keseluruhan pada perangkat yang dipantau serta pada gambar 25 adalah tampilan *load CPU* yang menampilkan *load average* atau beban kinerja pada CPU yang ditampilkan menjadi 3; yaitu 1 menit, 5 menit dan 15 menit, dari keseluruhan perangkat yang terpantau rata-rata penggunaan beban kinerja pada cpu sekitar 1.00 dalam waktu yang sama dengan pemantauan lainnya.

Gambar 24. Tampilan pemantauan CPU pada perangkat jaringan



Gambar 25. Tampilan pemantauan load CPU

Pada kolom lainnya, *librenms* menyediakan pemantauan penggunaan *Memory* pada seluruh perangkat jaringan yang terdaftar. Seperti pada gambar 26 yang menunjukkan grafik penggunaan *Memory* pada perangkat yang dipantau dalam rentang waktu per 6 jam, 24 jam, hingga tahunan pada setiap detail perangkat. Dari hasil pemantauan, rata-rata penggunaan memori sebanyak 30% pada keseluruhan perangkat sejumlah 130 MiB dari total 470 MiB.

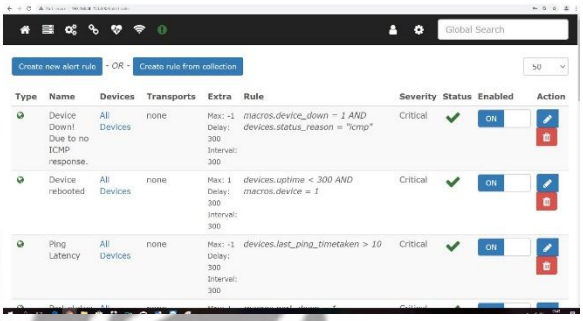
Gambar 26. Tampilan penggunaan Memory pada perangkat jaringan

Gambar 27. Tampilan pemantauan Suhu pada server

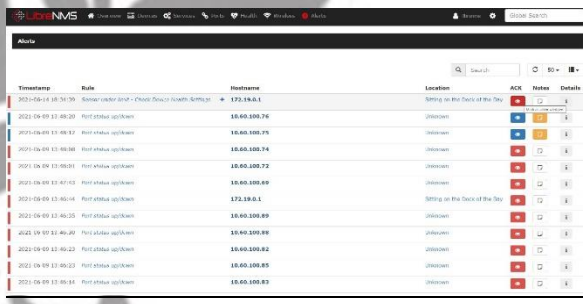
Selain melakukan pemantauan penggunaan *CPU* dan *Memory*, *librenms* dapat juga melakukan pemantauan kondisi suhu pada perangkat jaringan. Salah satunya pada gambar 27 merupakan kondisi suhu pada *server* yang saat ini digunakan. Terdapat detail temperatur pada setiap *core* yang terpantau, data yang diambil dalam kurun waktu harian menunjukkan kondisi suhu rata-rata berada di 40-45 derajat *Celcius*.

Adapun fitur pada *librenms* yaitu sistem peringatan yang didalamnya terdapat beberapa jenis peringatan berdasarkan *level*-nya. Sistem peringatan tersebut dapat dibuat berdasarkan *rule* yang sudah tersedia oleh *librenms* ataupun dapat dikustomisasi sesuai kebutuhan pengguna. pada gambar 28 merupakan menu untuk memilih *rule* yang akan diterapkan pada sistem peringatan.

Untuk Hasil sistem peringatan dapat dilihat pada gambar 29,terdapat peringatan yang menunjukkan kondisi pada perangkat jaringan terdapat kendala, diantaranya menampilkan bahwa status *port* pada perangkat mengalami *down* dalam kurun waktu singkat dan kemudian kembali *up*. Hal ini sangat memudahkan antara administrator dengan administrator lainnya untuk mengetahui kondisi pada perangkat jaringan secara bersamaan. Terdapat juga keterangan *acknowledge* sebagai penanda bahwa peringatan tersebut telah dibaca oleh administrator dengan tanda warna merah dan biru pada kolom kanan**.**



Gambar 28. Tampilan menambahkan rule yang dapat dikustomisasi sesuai kebutuhan



Gambar 29. Tampilan halaman alert

PENUTUP

Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan, Konfigurasi aplikasi *Librenms* yang berjalan pada *docker container* berhasil dilakukan dengan melakukan instalasi dan konfigurasi *docker container* dan membuat konfigurasi *docker compose* untuk menjalankan aplikasi *librenms* serta

menambahkan perangkat jaringan yang akan dimonitoring dengan mengkonfigurasi SNMP *agent* pada perangkat.

Aplikasi *Librenms* yang dijalankan menggunakan *docker* mampu melakukan pemantauan pada perangkat jaringan baik dari kondisi penggunaan trafik data maupun kondisi penggunaan *Resource* serta menunjukkan hasil pemantauan pada kondisi trafik data dengan rata-rata penggunaan pada pukul 07.30 sampai 14.00 yang disajikan dalam bentuk grafik dengan teknologi *Rrdtools*, serta *librenms* dapat mengkustomisasi *rule* pada sistem peringatan yang dapat mempermudah administrator dalam melakukan pemantauan jaringan dengan adanya fitur *alert history*.

Pemantauan penggunaan pada perangkat menunjukkan hasil penggunaan CPU dengan rata-rata 2%, *Load CPU* dengan rata-rata 1.00 dari keseluruhan, penggunaan *Memory* dengan rata-rata 30% pada seluruh perangkat jaringan yang terpantau. Hal ini menunjukkan bahwa aplikasi *librenms* di dalam *server* yang dijalankan menggunakan *Docker container* berjalan dengan kondisi normal tanpa adanya kendala. Dapat disimpulkan bahwa implementasi *network monitoring system* menggunakan aplikasi *Librenms* yang di *deploy* menggunakan *docker* berhasil dilakukan.

Saran

Aplikasi *Librenms* yang dijalankan pada *docker container* ini jauh dari kata sempurna, untuk kedepannya diharapkan dapat melakukan pengembangan yang lebih terhadap fitur yang telah disediakan agar mempermudah seorang administrator dalam melakukan pemantauan jaringan.

DAFTAR PUSTAKA

Apridayanti, S., Isnawaty, I., & Saputra, R. A. (2018). "*Desain Dan Implementasi Virtualisasi Berbasis Docker Untuk Deployment Aplkasi Web*". SemanTIK, 4(2), 37-46.

Docker, "*What is a Container? A standardized unit of software*". Docker. Retrieved From : https://www.docker.com/resources/what-container.

Harfiansyah, Irsyad. "*Mengenal Teknologi Docker*". Codepolitan. 18 Feb 2021. [Online]. Retrieved From : https://www.codepolitan.com/mengenal-teknologi-docker.

Hidayat, A., & Rizki, A. (2020). "*Monitor Jaringan Komputer Berbasis Web Menggunakan Cacti*". JUTEKIN (Jurnal Teknik Informatika), 8(1).

LibreNMS, "*Community-based gpl-licensed network monitoring system*". Librenms. 12 Jan 2021. [Online]. Retreive from: http://www.librenms.org/.

Muilwijk, Robin. "*Top 5 open source web servers*". opensource.com. 10 Jan 2021 [Online]. Retrieved From : https://opensource.com/business/16/8/top-5-open-source-web-servers.

Nugroho, Y. H., Sastra, N. P., & Wiharta, D. M. (2018). "*Analisis Unjuk Kerja Pemantauan Jaringan OpenNMS (Open Network Monitoring System) pada Jaringan TCP/IP*". Jurnal SPEKTRUM, 5(2), 158-166.

Pratama, M. R., Munadi, R., & Hafidudin, H. (2017). "*Implementasi Dan Analisis Sistem Monitoring Menggunakan Simple Network Management Protocol (snmp) Pada Gedung A, n, o Di Jaringan Telkom University*". eProceedings of Engineering, 4(2).

Saputra, I. W. K., Wiharta, D. M., & Sastra, N. P. (2020). "*Implementasi Sistem Pemantauan Jaringan Menggunakan Librenms Pada*

Jaringan Kampus Universitas Udayana". Jurnal SPEKTRUM, 7(2), 81-89.

Segara, A. P., Primananda, R., & Akbar, S. R. (2018). "*Implementasi MQTT (Message Queuing Telemetry Transport) pada Sistem Monitoring Jaringan berbasis SNMP (Simple Network Management Protocol)*". Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN, 2548, 964X.

Solehfuddin, M., Sugiyono, S., & Awaludin, M. (2016). "*Penerapan Simple Network Management Protocol Pada Fcaps Untuk Monitoring Server Berbasis Android Studi Kasus Pt Jaring Synergi Mandiri*". CKI ON SPOT, 9(2).

Yahya, A. S. (2021). "*Strategi Meningkatkan Produktivitas Kinerja Aparatur Sipil Negara Selama Work From Home Di Tengah Pandemi Covid-19: Mengubah Ancaman Menjadi Peluang. Tetap Kreatif Dan Inovatif Di Tengah Pandemi Covid-19*", 1, 62.